Information and Software Technology 150 (2022) 106966

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

On the effectiveness of testing sentiment analysis systems with metamorphic testing

Mingyue Jiang [a,*], Tsong Yueh Chen [b], Shuai Wang [c]

[a] School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China
[b] Department of Computer Science and Software Engineering, Swinburne University of Technology, Hawthorn, 3122, VIC, Australia
[c] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China

ARTICLE INFO

Keywords:
Metamorphic testing
Metamorphic relation
Sentiment analysis
False satisfaction

ABSTRACT

Context: Metamorphic testing (MT) has been successfully applied to a wide scope of software systems. In these applications, the testing results of MT form the basis for drawing conclusions about the target system's performance. Therefore, the effectiveness of MT is crucial to the trustfulness of the derived conclusions.
Objective: However, due to the nature of MT, its effectiveness can be affected by various factors. Despite of MT's success, it is still important to study its effectiveness under different application contexts.
Method: To investigate the effectiveness of MT, we focus on an important aspect, namely, false satisfactions (which are satisfactions of metamorphic relations that involve at least one failing execution), and revisit the application of MT to sentiment analysis (SA) systems. An in-depth analysis of the essence of false satisfactions reveals the situations where they would occur, and how they would affect the effectiveness of MT. Furthermore, 20 metamorphic relations (MRs) are identified for supporting a user-oriented evaluation of SA systems.
Results: The occurrence rates of false satisfactions are reported with respect to four SA systems. For the majority of MRs, false satisfactions account for about 20% to 50% of all MR satisfactions, suggesting that false satisfactions occur quite frequently in the evaluation of SA systems. It is also demonstrated that such high occurrence rates of false satisfactions adversely affect the users' selection of SA systems.
Conclusion: Our analysis reveals that without considering the occurrence of false satisfactions, MT may overestimate the system's conformance to the relevant MR. Furthermore, our experiments empirically show that conclusions derived from MT can be adversely affected when there are many false satisfactions. Our findings will help the MT community to adopt a more fair and reliable way of using the test outcomes of MT, and can also inspire the development of solid foundations for MT.

1. Introduction

Metamorphic testing (MT) [1–3] is a property based testing technique. The core component of MT is the metamorphic relations (MRs), which specify expected properties via the relationships among multiple inputs and outputs. Traditional testing techniques verify individual outputs of the system under test, where an oracle is required for determining the correctness of each output. Nevertheless, MT checks the relationships among a group of inputs and outputs against the relevant MR to see whether the MR is violated or satisfied, reporting MR violations or MR satisfactions as the testing results. Because of this, MT can be conducted in the absence of oracles, and hence has also been widely used for alleviating the oracle problem. Currently, MT has been successfully used in software verification and validation [4–11], quality assessment [12,13], system comprehension [14], software proving [15], and program repair [16,17]. Furthermore, MT has been recently applied to various artificial intelligence applications, such as computer vision [18] and NLP [19,20].

Despite of its success, there has been an increasing concern about the effectiveness of MT. That is, how accurately the evaluation results from MT can reflect the target system's real performance. When applying MT, although an MR violation indicates that at least one of the relevant outputs is incorrectly computed, an MR satisfaction is unable to tell whether or not all of the relevant outputs are correct. Notably, according to the intrinsic essence of MRs, an MR may still be satisfied even if some of the relevant outputs are incorrectly computed. Such MR satisfaction is referred to as the *false satisfaction* (FS) in this study. Obviously, a high occurrence rate of FS threatens MT's effectiveness, and further affects the trustfulness of the conclusions derived from MT. Having said that, MT faces the same limitation as traditional testing

* Corresponding author.
E-mail addresses: mjiang@zstu.edu.cn (M. Jiang), tychen@swin.edu.au (T.Y. Chen), shuaiw@cse.ust.hk (S. Wang).

https://doi.org/10.1016/j.infsof.2022.106966
Received 10 August 2021; Received in revised form 13 January 2022; Accepted 25 May 2022
Available online 2 June 2022

techniques that testing demonstrates the existences of faults but not the absence of faults, because MT is a testing method.

Although FS has been recognized by some researchers [21], none of the existing studies had analyzed the impacts of FS on the effectiveness of MT, nor had they investigated the causes leading to the high occurrence of FS. In fact, most of existing studies of MT are conducted under an implicit assumption that *false satisfactions rarely occur* [4,5,13,16]. We do agree that for most software systems, MT may have a very low chance of incurring FS and thus MT can be effective. However, it is still important to explore the situations where FS has a high occurrence rate and then to further investigate how FS affects MT's effectiveness in these situations. To this end, this study revisits the application of MT to sentiment analysis (SA) [22], which not only is a typical NLP task that has been a useful tool in various domains, but also has been observed to have quite high occurrence rates for FS according to our preliminary study.

Currently, SA has been evaluated by MT using a few MRs, revealing potential issues on some quality aspects, such as fairness [19,20] and robustness [23]. Differently, in this study, 20 MRs are designed from the users' perspective by considering different usage scenarios as well as different SA granularities. Moreover, to reveal the effectiveness of MT in testing SA systems, this study focuses on the following research questions.

- **RQ1:** How often do false satisfactions occur in testing SA?
- **RQ2:** How does the false satisfactions affect MT's effectiveness in testing SA?
- **RQ3:** What are the causes for the false satisfactions in testing SA?

The contributions of this study are summarized as follows.

- We identify a list of 20 MRs for SA. These MRs reflect the users' interests or preferences on the SA system, and thus their satisfaction rates reveal to what extent the system satisfies the users' specific needs. Accordingly, MT is conducted in a user-oriented way, and aims at revealing the degrees to which SA systems meet the users' preferences or expectations.
- We apply MT to evaluate four SA systems, the Amazon API, Microsoft API, Heroku API, and the Stanford SA package. We inspect the evaluation data to report the occurrence rates of FS, accordingly answering **RQ1**.
- We clarify the essence of FS and also reveal how FS affects the evaluation metrics of MT, to answer **RQ2**. In addition, we report a series of false satisfactions and also demonstrate how FS affects the users' selection of SA systems using MT, which empirically answers **RQ2**.
- We thoroughly investigate MRs as well as the characteristics of SA systems to uncover the causes behind the high occurrence rates of FS. This helps to answer **RQ3**.

The remainder of the paper is organized as follows. Section 2 introduces the technique of MT. Section 3 presents and analyzes the FS of MRs, and accordingly shows the impacts of FS on MT. Section 4 presents the application of MT to SA systems. Section 5 reports our experimental setup, and Section 6 presents experimental results to answer our research questions. Section 7 discusses some related work, and Section 8 concludes our study.

2. Metamorphic testing

Metamorphic testing (MT) [1,2] is a technique that is simple in concept but powerful in application. MT is well-known for its capability of alleviating the oracle problem and its ability of revealing program bugs [5–8,10,11,24]. These successes are mainly due to the core element of MT, namely, the metamorphic relation (MR). An MR describes a necessary property of the algorithm being implemented via the relationships among multiple inputs and outputs of the target program. In MT, an MR is used for generating test cases as well as for checking the testing results. A group of source and follow-up inputs of an MR is called a metamorphic group of inputs (MG) [1].

Consider a program P implementing the *sine* function. Because it is difficult or even impossible to determine the sine value for any arbitrary angle, testing of P faces the oracle problem. By considering some properties of the *sine* function, an MR can be "the sine value for an angle x should be equal to that for $\pi - x$". To apply this MR, any angle x can act as a *source input*, based on which a *follow-up input*, namely, $\pi - x$, can be generated. In this case, $(x, \pi - x)$ is an MG of the MR. Then, P is executed on both the source and follow-up inputs respectively, yielding the *source output* $P(x)$, and the *follow-up output* $P(\pi - x)$. Finally, $P(x)$ and $P(\pi - x)$ are checked against the MR: if they are not identical, the MR is said to be violated on this MG, which is thus called a violation revealing MG or simply a *violating* MG; otherwise, the MR is said to be satisfied on this MG, which is thus called a *non-violating* (or *satisfying*) MG. Violating MGs reveal bugs for P. When applying MT to test a program, multiple MGs will be used and the MR violation on any MG will indicate the existence of bugs. As a reminder, the other processes of MT can be easily automated once MRs are identified.

Although MT is originally proposed for the purpose of verification, it has been extended to support validation, quality assessment, and system comprehension. Xie et al. [4,13] applied MT to validate supervised and unsupervised machine learning systems, where the MR violations indicate that the relevant system is not suitable for that particular application scenario. Zhou et al. [12] validated online search services by using MT, and the satisfaction and violation of MRs help the users to better understand these services. They further proposed to apply MT to software with a diverse user base, where different users have different or even conflicting expectations on the system. MT-based validation of these systems can enhance the users' understanding about the system and then facilitate the selection and use of the right system services [14]. Notably, in these application scenarios, MRs can be used to describe the users' expectations on the target system and MR violations thus reveal the deviation of the system's behavior from the specified expectations.

Nowadays, MT has been employed to evaluate a broad range of systems, such as embedded systems [25], simulation systems [10,26], deep learning frameworks [27,28], machine translators [29,30], the fairness of sentiment analysis models [19,20], etc. Apart from that, MT has also been integrated into various other areas beyond testing, such as program proving [15] and program repair [16].

3. False satisfactions of metamorphic relations

This section reports an in-depth analysis of the false satisfaction of MRs. Two evaluation metrics in MT, namely, the violation rate (VR) and the satisfaction rate (SR), are investigated, based on which the impacts of false satisfaction on the effectiveness of MT are discussed and concluded.

3.1. False satisfaction

When a test case is used to test a system with respect to an oracle, it is *passed* (*failed*) if its real output is consistent (inconsistent) with the oracle. Obviously, the *pass* (✓) or *fail* (×) of a test case can be regarded as the truth test result of the test case in such situations. It is noted that the truth test result is attainable only if test oracles are available. However, MT is usually conducted without oracles, and thus the truth results of individual test cases are not available. Consequently, the test result reported by MT may not necessarily be consistent with the truth test result.

Let (t_s, t_f) be an MG for an MR, where t_s and t_f are its source and follow-up inputs, respectively. MT reports either a *satisfaction* or a *violation* by checking the relationships among the outputs of t_s and t_f against the MR. Generally, there can be two types of MR violations and also two types of MR satisfactions, which are summarized in Table 1 and are also detailed as below.

Table 1
The truth results of test cases vs. MT results.

MG	MT results	
(t_s, t_f)	Satisfaction	Violation
(✓,✓)	True	False
(✓, ×)	**False**	True
(×, ✓)	**False**	True
(×, ×)	**False**	True

- *True Satisfaction*: The MR is satisfied and both t_s and t_f are passed.
- *True Violation*: The MR is violated and at least one of t_s and t_f is failed.
- ***False Satisfaction***: The MR is satisfied but at least one of t_s and t_f is failed.
- *False Violation*: The MR is violated but both t_s and t_f are passed.

Obviously, *false violation* occurs only if the relevant MR incorrectly specifies the system's expected properties, and such an MR should not be selected for MT. That is, the number of *false violations* must be 0 as long as the MR is valid. In this study, we exclude the situation of having invalid MRs. Hence, in the MT evaluation, every MR violation is a *true violation*, which involves the execution of at least one failed test case. This further ensures the common interpretation that an MR violation is an indicator of bugs or the deviation of the system's behavior from the specified properties, suggesting that MT can be a feasible testing method in the absence of test oracles.

However, the situation is totally different for MR satisfactions. As shown in Table 1, although an MR satisfaction is expected from an MG consisting of merely passed test cases, it may not necessarily involve only passed test cases. Instead of being an indicator of the compliance of the system to the MR, a *false satisfaction* (FS) should indeed indicate the inconsistency between the system's behavior and the MR.

3.2. Violation rate and satisfaction rate in the presence of false satisfaction

VR and SR are two commonly used evaluation metrics in MT studies. VR measures the degree to which a system violates the relevant MR, while SR measures the degree to which a system satisfies the relevant MR. Formally, VR is defined as the ratio of the number of violating MGs over the total number of MGs; and SR is defined as the ratio of the number of non-violating MGs over the total number of MGs. These two metrics are complementary with each other such that $VR+SR = 1$.

At first, we follow the conventional practice to calculate VR and SR values based purely on the observed testing results of MT, that is, the numbers of MR violations and satisfactions observed after conducting MT. In this situation, we use the terms of *observed* VR and SR (denoted as VR_o and SR_o) to respectively represent these two values. Suppose that n MGs are used to evaluate a system, yielding n_s MR satisfactions and n_v MR violations, where $n > 0, n_s \geq 0, n_v \geq 0$, and $n_s + n_v = n$. Then, the VR_o is calculated as the ratio of n_v to n, and the SR_o is calculated as the ratio of n_s to n, which are depicted in the second row of Table 2. The VR_o or SR_o values are the raw evaluation results of MT: on one hand, $VR_o > 0$ ($SR_o < 1$) suggests that the system under test does not satisfy the MR, revealing defects in the system or the deviations of the system's behaviors from the expected properties. On the other hand, $VR_o = 0$ ($SR_o = 1$) is treated as an indicator of the satisfaction of the MR with respect to the used test data, suggesting that the system is likely to perform well with respect to the MR over all possible inputs. It should be noted that the calculations of VR_o and SR_o totally ignore the existence of FS.

We now consider the existence of FS. In this situation, we use the terms of *genuine* VR and SR, which are denoted as VR_g and SR_g, respectively. Suppose that among n_s MR satisfactions, there are n_{fs} false satisfactions, where $0 \leq n_{fs} \leq n_s$. As a reminder, the existence of n_{fs} FS suggests that there must be n_{fs} MR violations in addition to the n_v observed MR violations, while there must be n_{fs} MR satisfactions being excluded from the n_s observed MR satisfactions. Accordingly, the genuine numbers of MR violations and satisfactions are $(n_v + n_{fs})$ and $(n_s - n_{fs})$, respectively. As a result, VR_g and SR_g are calculated in a different way from VR_o and SR_o, as shown in the third row of Table 2.

Table 2
The observed VR and SR values (VR_o and SR_o which are calculated without considering FS) and the genuine VR and SR values (VR_g and SR_g which are calculated by taking FS into account).

Violation rate (VR)	Satisfaction rate (SR)
VR_o: $\frac{n_v}{n_v+n_s}$	SR_o: $\frac{n_s}{n_v+n_s}$
VR_g: $\frac{n_v+n_{fs}}{n_v+n_s}$	SR_g: $\frac{n_s-n_{fs}}{n_v+n_s}$
$VR_g = VR_o + SR_oR_{FS}$ $SR_g = SR_o(1 - R_{FS})$	

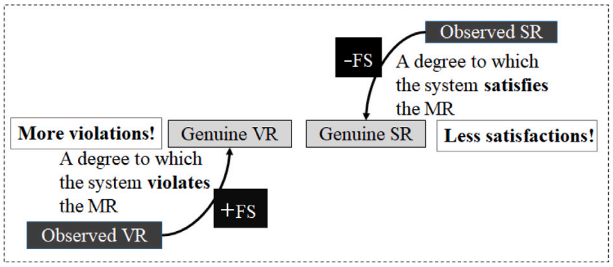
Observed SR
-FS
A degree to which the system **satisfies** the MR
More violations!
Genuine VR
Genuine SR
Less satisfactions!
A degree to which the system **violates** the MR
+FS
Observed VR

Fig. 1. Both observed VR and SR optimistically overestimate the system's compliance with the MR.

Let R_{FS} be the FS occurrence rate, that is, $R_{FS} = {}^{n_{fs}}/_{n_s}$. A higher R_{FS} value indicates a more frequent occurrence of FS. The relationship between VR_g and VR_o can be expressed as $VR_g = VR_o + SR_oR_{FS}$. Similarly, it can also be inferred that $SR_g = SR_o(1 - R_{FS})$.

Obviously, the VR_o (SR_o) value may be different to the corresponding VR_g (SR_g) value. Moreover, the more false satisfactions are, the larger significant discrepancies between them will be. Because the existence of FS implies more MR violations and less MR satisfactions, the VR_g value should be larger while the SR_g value should be smaller as compared with the VR_o and SR_o values, as depicted in Fig. 1. Consequently, both the observed VR and SR optimistically overestimate the system's compliance with the MR, which will threaten the trustfulness of conclusions derived from MT.

In summary, in view of the existence of FS in MT evaluation, we make the following conclusions.

- Either $VR_o > 0$ or $SR_o < 1$ is a definite indicator of successfully detecting the deviation of the system's behavior from the properties specified by the relevant MR. This is because every observed MR violation is a genuine violation.
- $VR_o = 0$, or equivalently $SR_o = 1$ does not guarantee the full compliance with the MR. The reason for this is the possible occurrence of FS.
- Although identifying FS amongst observed MR satisfactions is challenging, it is still critical to know the reason and frequency of FS occurrence in order to better use the testing results of MT.

4. Metamorphic testing of sentiment analysis systems

To empirically investigate false satisfactions of MRs and then answer our research questions, we must apply MT to systems where false satisfactions are not rare. Therefore, we propose to apply MT to sentiment analysis. In this section, we first introduce the task of sentiment analysis. Then, we specify our MRs for sentiment analysis.

4.1. Sentiment analysis

Nowadays, there are numerous platforms, such as forums, websites, and blogs, for users to express their opinions on any topic. Hence, exploring the perspectives of users from these opinionated information is of significant importance. Sentiment analysis (SA) [22], which is also known as opinion mining, leverages the techniques of natural language processing, text analysis, and computational linguistics to extract or identify sentiment information in a textual data. The sentiment information, which is generally classified as *positive*, *negative*, or *neutral*, describes people's opinions, attitudes and emotions towards an entity (such as a topic, an event, a service, an individual, etc.).

SA has been widely applied for extracting sentiments or opinions from social media [31] to support various commercial activities as well as political and social events. On one hand, SA is a useful tool for sale trend prediction [32,33], stock market prediction [34], advertisement recommendation [35], etc. On the other hand, it also supports political events (such as election [31]), and is helpful for discovering important information about social events (for example, fighting infectious diseases [36] and security attack prediction [37]). More recently, SA has been applied to scientific citations to demonstrate its effectiveness in summarizing the attitudes of authors towards cited papers [38]. Despite of the importance and prevalent application of SA, existing SA systems still confront several issues. SA has been continuously reported to be biased towards certain races and genders [19,20,39]. Moreover, it has been demonstrated to be of weak robustness in handling various linguistic phenomena. For example, SA might be fooled by some adjectives [40], and it was sensitive to typos and some type of contents (such as URLs [23] and words that are not in the data set [41]). Therefore, it is important to properly evaluate SA systems in order to get a deeper insight about its capabilities and weaknesses.

Determining the sentiment polarity of a sentence or document is non-trivial. Sentiment can be expressed in different manners. For example, a sentence conveying a *negative* sentiment may not use any negative words. More importantly, sentiment is always context-sensitive and domain-dependent. For instance, in a movie review, some bad parts of the movie (for example, a bad character) should not make the movie bad; however, in a product review, some bad parts of the product should affect the reputation of the product. This increases the difficulty of confirming the sentiment of a text. Moreover, since SA has a diverse user base, the differences in users' knowledge backgrounds, needs, preferences and goals cause discrepancies in users' understanding and expectations on the SA results. All of these imply that evaluating SA has the oracle problem [42].

MT has been applied to SA systems to evaluate their fairness [19, 20], robustness and some other properties [23]. Differently, in this study, we evaluate SA systems from the users' perspective by considering different usage scenarios as well as different SA granularities. To this end, we design MRs that reflect the users' interests or preferences on SA systems, and thus their satisfactions or violations reveal to what extent the system satisfies the users' specific needs. Consequently, the evaluation results can help the user to better understand the systems' capability and then to make better use of them.

4.2. Definitions and concepts

SA can be conducted at three different levels in terms of the granularity, that is, document-level, sentence-level and aspect-level [43], where the former two respectively take a whole document and a sentence as the basic information unit, while the last one focuses on identifying the sentiment with respect to the specific aspects of the relevant entity. This study focuses only on the sentence-level and document-level SA.[1]

Input and output of a SA system: The input to the sentence-level SA is a sentence, and the input to the document-level SA is a document containing multiple sentences. The output of a SA system is $(l,\ p)$, that is, a label l ($l \in \{positive, negative, neutral\}$) accompanied with a confidence score p ($0 \leq p \leq 1$).[2]

Perturbing a sentence or a document: Perturbations refer to operations used to change a sentence or a document. In this study, each MR involves one specific type of perturbation, which, after being applied on a source input, yields a follow-up input. Since a perturbation introduces a small change to an input, it is easier to identify changes of the relevant output, if any.

Relationships among SA outputs: Suppose O_1 and O_2 are two outputs of a given SA system. They can have the following relationships.

- *Equivalent sentiments:* The sentiments conveyed by O_1 and O_2 are regarded to be equivalent if $O_1.l = O_2.l$.
- *Totally different sentiments:* The sentiments conveyed by O_1 and O_2 are regarded to be totally different if $O_1.l \neq O_2.l$.
- *A stronger sentiment:* The sentiment of O_2 is regarded to be stronger than that of O_1 if $O_2.l = O_1.l$ and $O_2.p > O_1.p$.
- *Sentiments keep going up:* Sentiments conveyed by a list of outputs $O_1, O_2, \ldots, O_n$ ($n \geq 3$) are regarded to keep going up if $\forall\ i, 1 < i \leq n$, either of the following conditions holds:
  (i) $O_{i-1}.l \neq O_i.l$, and ($O_{i-1} = negative$ and $O_i = neutral$, or $O_{i-1} = neutral$ and $O_i = positive$, or $O_{i-1} = negative$ and $O_i = positive$);
  (ii) $O_{i-1}.l = O_i.l = positive$, and $O_{i-1}.p \leq O_i.p$;
  (iii) $O_{i-1}.l = O_i.l = negative$, and $O_{i-1}.p \geq O_i.p$;
  (iv) $O_{i-1}.l = O_i.l = neutral$.
- *Sentiments keep going down:* Sentiments conveyed by a list of outputs $O_1, O_2, \ldots, O_n$ ($n \geq 3$) are regarded to keep going down if $\forall\ i, 1 < i \leq n$, either of the following conditions holds:
  (i) $O_{i-1}.l \neq O_i.l$, and ($O_{i-1} = positive$ and $O_i = neutral$, or $O_{i-1} = neutral$ and $O_i = negative$, or $O_{i-1} = positive$ and $O_i = negative$);
  (ii) $O_{i-1}.l = O_i.l = positive$, and $O_{i-1}.p \geq O_i.p$;
  (iii) $O_{i-1}.l = O_i.l = negative$, and $O_{i-1}.p \leq O_i.p$;
  (iv) $O_{i-1}.l = O_i.l = neutral$.

4.3. MRs for sentiment analysis

In order to support the evaluation of SA systems in various domains, it is important to use MRs that involve the basic nature of SA without reference to the specific knowledge of any application domains. Moreover, we aim to have the used MRs to cover various types of output relationships as explained in Section 4.2. As a reminder, an MR not only specifies the way of perturbing source inputs to generate the relevant follow-up inputs, but also describes the expected relationships among source and follow-up outputs.

We identified two different sets of MRs for sentence-level SA and document-level SA, respectively. Each MR applies one type of perturbation on the source input I_s to construct the follow-up input I_f, and further expresses the users' expectations on the changes of sentiment caused by the perturbation, namely, the relationship between the source output R_s and the follow-up output R_f. For an MR, a pair of (I_s, I_f) form an MG. In this study, we designed MRs involving exactly one source input and one follow-up input, and also those involving one source input and multiple follow-up inputs.

[1] We did not consider the aspect-based SA in this study, because it is essentially different from the sentence-level and document-level SA: it determines sentiments for different aspects of an entity, rather than extracting an overall sentiment of a text.

[2] For some SA systems providing only a sentiment label, the corresponding confidence score is regarded to be a constant.

Table 3
MRs for sentence-level SA: interpretations of MR satisfaction and illustrative input examples.

MR	Interpretation of MR satisfaction	Example MG (I_s, I_f)
MR1.1	SA treats abbreviations or contractions and full forms in a similar way.	("don't dismiss this film because of its sources.", "do not dismiss this film because of its sources.")
MR1.2	SA is capable of handling different synonyms.	("I want to buy a bike.", "I want to buy a bicycle.")
MR1.3	SA treats singulars and plurals in a similar way.	("I buy a book.", "I buy some books.")
MR1.4	SA treats '!' as a form of emphasis.	("It's a great day.", "It's a great day!")
MR2.1	SA is capable of handling different emphasizing adverbs.	("The apple is sweet.", "The apple is very sweet.")
MR2.2	SA is capable of handling negations.	("The weather is bad.", "The weather is not bad.")
MR3.1	SA is insensitive to the cases of words.	("it'll make sense when you see it.", "IT'LL MAKE SENSE WHEN YOU SEE IT.")
MR3.2	SA treats the capitalized words as a form of emphasis.	("The flower is beautiful.", "The FLOWER is BEAUTIFUL.")
MR4.1	SA is insensitive to the tense of the sentence.	("The storyline is as cliched as they come.", "The storyline was as cliched as they come.")
MR4.2	SA is insensitive to the order of words connected by some conjunctions.	("The film is wonderful and exciting.", "The film is exciting and wonderful.")
MR4.3	SA is insensitive to different conjunctions.	("Although I like it, I will not buy it.", "I like it, but I will not buy it.")
MR4.4	SA is insensitive to different comparative descriptions.	("The car is faster than the bicycle.", "The bicycle is slower than the car.")
MR4.5	SA is sensitive to the orders of comparative objects.	("The car is faster than the bicycle.", "The bicycle is faster than the car.")

4.3.1. *MRs for sentence-level SA*

When a sentence is provided to SA, the sentiment polarity of this sentence is determined. It is noted that a sentiment can be expressed in varying and subtle ways. Moreover, sentences provided to SA are always written by ordinary users rather than seasoned writers or professional journalists, which are thus of different styles and forms. Therefore, SA is expected to be able to handle sentences written in varying forms and styles.

To explore SA's capability of analyzing individual sentences, we designed a series of MRs, which consider the users' common practice of writing and altering sentences in different usage scenarios, and also take into account the users' common expectations on the capabilities of SA. As a reminder, these MRs are identified from the users' perspective. Therefore, the satisfaction of an MR on some MGs suggests that the SA system under investigation satisfies the users' expectations to a certain degree, and the violation of an MR reflects the deviation of the system's specific capability with respect to the MR, rather than revealing the system's defects.

We classify all MRs into four categories based on the perturbation operations involved. In the following, each MR is detailed and explained, and the corresponding interpretation of MR satisfaction and illustrative MGs are shown in Table 3.

(1) Replacing some words or phrases with others. Given a sentence, it is probable to rephrase the sentence by replacing some words or phrases with the relevant words or phrases. This category of MRs is designed to investigate how SA responds to different forms of words or phrases which expresses similar meanings.

- *MR1.1 (abbreviations or contractions ↔ full forms):* We construct I_f by replacing abbreviations or contractions in I_s with their corresponding full forms (or vise versa). Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR1.2 (nouns ↔ synonyms):* We construct I_f by replacing some nouns in I_s with their synonyms. Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR1.3 (singulars ↔ plurals):* We construct I_f by replacing singulars in I_s with their corresponding plurals (or vise versa). Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR1.4 ('.' ↔ '!'):* Suppose that R_s is either *positive* or *negative*. We construct I_f by replacing the period with the exclamatory mark. Then, R_f is expected to have a *stronger* sentiment than R_s.

MR1.1 and MR1.2 focus on the users' writing preferences: the users may or may not prefer to use contractions, and they may be used to utilize words they are familiar with. MR1.3 further considers the use of singulars and plurals. For these three MRs, the replacement is expected not to affect the sentiment significantly, because the meanings of sentences before and after perturbation are almost identical. MR1.4 aims at investigating the impact of different punctuations on the sentiment. Because the users would use an exclamatory mark for the purpose of emphasis, using an exclamatory mark as a replacement for a period is expected to increase the intensity of the sentence's sentiment.

(2) Adding some words. A sentence can be slightly altered by adding some words, but leading to changes to its sentiment. This category of MRs purposely adds some words into a sentence in order to check whether or not SA can properly reflect the impact of the added words on the sentiment.

- *MR2.1 (Adding emphasizing adverbs):* Suppose that R_s is either *positive* or *negative*. We construct I_f by adding some emphasizing adverbs into I_s. Then, R_f is expected to have a *stronger* sentiment than R_s.
- *MR2.2 (Adding negation words):* Suppose that R_s is either *positive* or *negative*. We construct I_f by adding some negation words into I_s. Then, R_f is expected to have a *totally different* sentiment from R_s.

Users would use some words, such as 'very', 'extremely', and 'absolutely', for the purpose of emphasis. MR2.1 focuses on this usage, and can be used to check whether SA is sensitive to these emphasizing adverbs. MR2.2 investigates whether or not the negation reversing the sentence's meaning will significantly affect the sentiment.

(3) Capitalizing words. In English, capitals can be used in different ways to serve for different purposes. This category of MRs is designed to investigate how SA deals with capitals.

- *MR3.1 (Making all characters upper-case/lower-case):* We construct I_f by reversing the case of all characters of I_s. Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR3.2 (Making nouns and adjectives upper-case):* Suppose that R_s is either *positive* or *negative*. We construct I_f by making all nouns and adjectives of I_s in upper-case. Then, R_f is expected to have a *stronger* sentiment than R_s.

MR3.1 aims to investigate the robustness of SA in terms of the cases of characters. However, MR3.2 is designed to inspect whether or not SA follows one usage of capitals, that is, using words in capital as a kind of emphasis.

(4) Restructuring the sentence. Different users may write different sentences to express the same meaning or to describe the same object. This category of MRs is designed to investigate whether SA is fair to sentences written in different styles.

- *MR4.1 (Changing the tense):* We construct I_f by altering the tense of I_s. Then, R_f is expected to have an *equivalent* sentiment with R_s.
- *MR4.2 (Swapping some words or phrases connected by 'and', 'or', 'either...or', and 'be different from'):* We construct I_f by swapping some words or phrases of I_s. Then, R_f is expected to have an *equivalent* sentiment with R_s.
- *MR4.3 (Switching between 'although' and 'but' to connect clauses):* We construct I_f by using a different conjunction from I_s. Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR4.4 (Rephrasing comparative sentence):* We construct I_f by rephrasing I_s (with the form '[noun1] is [c1] than [noun2]') as '[noun2] is [c2] than [noun1]', where c1 is a comparative word and c2 is the antonym of c1. Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR4.5 (Swapping comparative objects):* Suppose that R_s is either *positive* or *negative*. We construct I_f by swapping comparative objects in I_s, that is, by changing I_s (with the form '[noun1] is [c] than [noun2]') to '[noun2] is [c] than [noun1]', where c is a comparative word. Then, R_f is expected to have a *totally different* sentiment with R_s.

MR4.1 inspects whether or not SA is sensitive to the tense of the sentence. MR4.3 and MR4.4 are used to measure the SA's capability of handling different forms of sentences. However, MR4.2 and MR4.5 investigate to what extent the SA is order dependent.

4.3.2. MRs for document-level SA

A document consists of one or multiple sentences. When a document is provided as input, SA treats the whole document as an information unit and determines the sentiment expressed by the whole document. Noted that all MRs defined in Section 4.3.1 can also be applied to document-level SA by perturbing some sentences of the document. We further identify some MRs that are specific for document-level SA.

(1) Permuting sentences of a document. This category of MRs is designed for assessing the impact of sentences' order on the SA result.

- *MR5.1 (Random permutation):* I_f is constructed by changing the order of sentences in I_s. Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR5.2 (Grouping sentences with the same sentiment label together):* I_f is constructed by changing the orders of sentences of I_s such that sentences with the same sentiment label are placed together. Then, R_f is expected to have an *equivalent* sentiment to R_s.
- *MR5.3 (Grouping sentences according to both the sentiment label and confidence score):* I_f is constructed by classifying sentences of I_s into three groups according to their sentiment labels, and then putting sentences of individual group in ascending or descending order by their confidence scores. Then, R_f is expected to have an *equivalent* sentiment to R_s.

(2) Adding sentences into a document. The following MRs are designed to investigate how additional sentences affect the overall sentiment of a document.

- *MR6.1 (Adding positive sentences gradually):* Suppose that $\{s_1, s_2, \ldots, s_n\}$ $(n \geq 2)$ is a set of sentences conveying *positive* sentiment. A set of follow-up inputs $\{I_f^1, I_f^2, \ldots, I_f^n\}$ are constructed such that $I_f^i = I_s \cup \{s_1, \ldots, s_i\}$ $(1 \leq i \leq n)$. Then, the series of output $R_s, R_f^1, \ldots, R_f^n$ convey *sentiments that keep going up*.
- *MR6.2 (Adding negative sentences gradually):* Suppose that $\{s_1, s_2, \ldots, s_n\}$ $(n \geq 2)$ is a set of sentences conveying *negative* sentiment. A set of follow-up inputs $\{I_f^1, I_f^2, \ldots, I_f^n\}$ are constructed such that $I_f^i = I_s \cup \{s_1, \ldots, s_i\}$ $(1 \leq i \leq n)$. Then, the series of output $R_s, R_f^1, \ldots, R_f^n$ convey *sentiments that keep going down*.

(3) Removing sentences from a document. This category of MRs is to check how the sentiment changes after removing some contents from a document.

- *MR7.1 (Removing all sentences having the same sentiment label as the document):* Suppose that I_s contains more than one sentences and R_s is either *positive* or *negative*. I_f is constructed from I_s by removing all sentences whose sentiment labels are identical to that of I_s. Then, R_f is expected to have a *totally* different sentiment from R_s.
- *MR7.2 (Removing all sentences having different sentiment labels from the document):* Suppose that I_s contains more than one sentences and R_s is either *positive* or *negative*. I_f is constructed from I_s by removing all sentences whose sentiment labels are different from that of I_s. Then, R_f is expected to have an *equivalent* sentiment to R_s.

5. Experimental setup

We conducted empirical experiments to assess the effectiveness of MT in evaluating SA systems, with the aim of empirically answering our three research questions. We apply MT to test four subject SA systems and identify FS of MRs (to address RQ1 regarding the occurrence rate of FS), and further report how FS affects the users' selection of SA systems (to address RQ2 regarding the impact of FS on the effectiveness of MT). Finally, we conduct a further investigation to uncover the causes behind the high occurrence rates of FS (to address RQ3 regarding the factors affecting the occurrence rate of FS). In the following, we first introduce the subject SA systems, and then explain the automation of MT and the way of preparing MGs for MRs.

5.1. SA systems under investigation

Our experiments involved four SA systems: the Amazon Comprehend API,[3] the Microsoft Azure API,[4] the Heroku API,[5] and the SA service provided by the Stanford coreNLP package.[6] The Amazon and Microsoft APIs provide commercial SA services, while Heroku is a free sentiment analysis API. We wrote Python scripts for using the remote

[3] https://aws.amazon.com/comprehend/.
[4] https://azure.microsoft.com/en-us/.
[5] https://sentim-api.herokuapp.com.
[6] https://stanfordnlp.github.io/CoreNLP/index.html.

services provided by these APIs. The Stanford coreNLP is an open source package. We installed it and then interacted with it via Python scripts.

The three APIs, namely, Amazon, Microsoft and Heroku, report the sentiment with both a polarity label and a score, while the Stanford SA package returns only a polarity label (it also has two additional labels: *very positive* and *very negative*). Moreover, the former three systems support both the sentence-level and document-level SA, but the Stanford SA package can only be used for the sentence-level SA.

5.2. Automation of MT

The general procedure of applying MT to SA is as follows. When a set of source inputs is prepared and an MR is identified, the corresponding follow-up inputs will be constructed according to the MR, resulting in a set of MGs. Then, both source and follow-up inputs of individual MGs are executed on the target SA system, yielding the source and follow-up outputs. Finally, source and follow-up outputs of each MG are checked against the MR, reporting an MR violation or MR satisfaction. Obviously, once the MR is known, both the construction of the follow-up inputs and the checking of output relationships can be automated.

In this study, we developed Python programs for constructing follow-up inputs and validating the relationships among relevant source and follow-up outputs with respect to each of the MRs specified in Section 4.3. In our implementations, Stanford parser[7] was employed to conduct the POS-tagging on a sentence in order to recognize nouns, verbs and adjectives, and the synonyms were searched via an online synonym dictionary.[8] Some basic string operations, including exchanging between lower case and upper case characters, substring matching, and substring substitution, were also utilized. Furthermore, we predefined a list of abbreviations and their normal forms, and also a list of emphasizing adverbs for implementing input operations of some MRs (e.g., MR1.1 and MR2.1).

5.3. Construction of MGs

All of the 20 MRs, which are specified in Section 4.3, were used in the experiments. Due to the intrinsic capabilities of our subject SA systems, it is noted that some MRs are not applicable to some of the SA systems (e.g., all of the document-level MRs cannot be applied to the Stanford SA package because this system does not support document-level SA).

To prepare source inputs for every of the MRs, we used the polarity dataset v2.0[9] of the Movie Review dataset [44]. This dataset consists of 2000 review files, each of which contains multiple sentences. We extracted sentences from individual files, each of which act as a candidate input for sentence-level MRs. Then, for a sentence-level MR, each candidate input is analyzed to determine whether or not it can be a source input for this MR. That is, a candidate input is a source input of the MR if the perturbation specified by the MR can be applied to this input. In the document-level evaluation, each review file is treated as a candidate input, which is further analyzed with respect to individual MRs in order to check whether or not it can be a source input of the relevant MR.

As described in Section 4.3, different MRs involve different perturbations. Moreover, some MRs have restrictions on the source outputs, which may be different for different SA systems. Therefore, different MRs usually have varying numbers of MGs, and an MR may also have different numbers of eligible MGs for different SA systems. In total, 279,473 MGs are applied to Amazon API, 316,120 MGs are used for Microsoft API, 319,520 MGs are applied to Heroku API, and 189,731 MGs are used for evaluating Stanford SA package.

6. Results and analysis

In this section, we analyze our experimental results in order to answer the three research questions.

6.1. The occurrence rate of FS (RQ1)

We first report the observed SR values of every MR with respect to individual subject systems. Then, we launch experiments to identify FS in the SA evaluation, and report the occurrence rates of FS.

6.1.1. Results of observed SR

Table 4 reports the observed SR values of every SA system. Each cell in Table 4 contains a SR_o value of a SA system with respect to a particular MR. Due to the intrinsic capabilities of SA systems, some MRs are not applicable to some of the SA systems, and thus the corresponding cells are annotated with '-'. In this table, the highest and the lowest SR_o values are 100% and 0%, respectively. For a SA system and an MR, $SR_o = 100\%$ means that no violation has been detected for this system with respect to all constructed MGs of this MR. Instead, $SR_o = 0\%$ means that all used MGs of the MR are violated by the system in the experiments, and thus the system shows great deviations from the properties specified by the MR. It is also observed from Table 4 that the majority of SR values are less than 100%, indicating that most of systems do not fully comply with the corresponding MRs.

Overall, the results show two observations reported as below.

- **A SA system has varying degrees of satisfaction for different MRs.** As shown in Table 4, each SA system has different SR_o values for all of the applicable MRs. For example, the SR_o values of Amazon API range from 24.28% to 99.92%; the SR_o values of Microsoft API range from 9.09% to 100%; the SR_o values of Heroku API range from 0% to 100%; and the SR_o values of Stanford package range from 14.29% to 94.21%. The discrepancies among the SR_o values of a system indicate that the system show varying performances for different MRs. Meanwhile, these discrepancies also reveal that a system is of varying robustnesses to different input changes specified by different MRs. For example, consider the application of the two sentence-level MRs, namely, MR1.1 and MR2.2, to the Microsoft API. The results suggest that this system does quite well with respect to MR1.1 (with a $SR_o = 98.38\%$) but has a poor performance with respect to MR2.2 (with a $SR_o = 20.44\%$). Obviously, compared with the input change of adding negative words (that is, the change introduced into the follow-up sentences of MR2.2), this system is more robust to the change of exchanging between abbreviations and full forms (which is the change introduced into the follow-up sentence of MR1.1).
- **Different SA systems satisfy an MR to different extents.** In our experiments, all SA systems have different SR_o values for every of the 20 MRs. For some MRs, the differences among SR_o values of different SA systems are very significant. Consider, for example, the SR_o values of MR3.1 on the four systems are 46.82%, 99.99%, 99.99% and 35.98%. These results indicate that the four SA systems behave very differently with respect to the relevant MR, and thus show varying degrees of satisfaction for the MR.

6.1.2. False satisfactions of individual MRs

In the SA evaluation, there is no available oracle information, and thus it is not possible to fully automate the procedure of identifying FS. Therefore, we conducted a hybrid checking process as below. For the document-level MRs that involve the *equivalent* output relationship, their non-violating MGs are automatically inspected by using the

[7] https://nlp.stanford.edu/software/lex-parser.shtml.
[8] https://www.thesaurus.com/.
[9] http://www.cs.cornell.edu/people/pabo/movie-review-data/.

Table 4
The observed SR values of the four SA systems with respect to different MRs.

MRs	Amazon	Microsoft	Heroku	Stanford
MR1.1	92.63%	98.38%	95.93%	94.21%
MR1.2	88.06%	92.05%	97.14%	89.06%
MR1.3	92.17%	95.93%	98.99%	86.13%
MR1.4	67.00%	34.27%	65.58%	–
MR2.1	70.25%	63.75%	10.43%	–
MR2.2	39.26%	20.44%	11.59%	27.27%
MR3.1	46.82%	99.99%	99.99%	35.98%
MR3.2	24.28%	34.20%	1.73%	–
MR4.1	93.70%	96.02%	99.23%	90.89%
MR4.2	94.84%	97.49%	99.41%	90.64%
MR4.3	80.89%	90.24%	99.98%	76.19%
MR4.4	27.27%	27.27%	54.55%	36.36%
MR4.5	28.57%	9.09%	0%	14.29%
MR5.1	61.80%	99.94%	93.47%	–
MR5.2	35.37%	99.75%	87.25%	–
MR5.3	34.63%	99.69%	84.72%	–
MR6.1	99.92%	68.18%	70.82%	–
MR6.2	99.92%	9.09%	65.75%	–
MR7.1	75.60%	100%	99.36%	–
MR7.2	99.04%	100%	100%	–

Table 5
FS occurrence rates of individual MRs.

MRs	Amazon	Microsoft	Heroku	Stanford
MR1.1	12.61%	28.89%	25.0%	29.06%
MR1.2	15.06%	20.92%	31.75%	36.97%
MR1.3	11.86%	31.31%	33.18%	42.08%
MR1.4	40.0%	10.32%	23.91%	–
MR2.1	28.23%	25.93%	50.0%	–
MR2.2	18.26%	45.35%	42.86%	2.9%
MR3.1	13.39%	26.58%	29.38%	28.77%
MR3.2	35.90%	20.96%	12.50%	–
MR4.1	15.97%	37.50%	19.41%	31.29%
MR4.2	12.24%	26.92%	41.82%	8.51%
MR4.3	27.59%	38.46%	25.0%	40.74%
MR4.4	0.0%	66.70%	66.70%	100%
MR4.5	0%	100%	0.0%	100%
MR5.1	45.96%	98.71%	41.99%	–
MR5.2	30.84%	98.76%	41.67%	–
MR5.3	29.36%	98.83%	41.02%	–
MR6.1	48.91%	0.0%	47.04%	–
MR6.2	48.89%	0.0%	38.83%	–
MR7.1	29.21%	0.0%	41.42%	–
MR7.2	29.34%	0.0%	41.50%	–

ground truth information of source inputs provided by the dataset. However, for the other document-level MRs and the sentence-level MRs, their non-violating MGs are checked in a manual way.

Due to the tremendous number of MR satisfactions (i.e., 735,561 in total), it is very expensive to manually inspect all data. Hence, for each MR and each system, we randomly sampled 1% MR satisfactions for manual checking. For an MR satisfaction, its related information, including the source and follow-up inputs and their outputs provided by the relevant SA system, were collected and inspected. Due to the fact that all inputs were originated from the Movie Review dataset [44], each of them describes the information related to some movies. Accordingly, we judged the sentiment of an input sentence or document by interpreting its meaning or implication so as to check whether or not it expresses something that is good or bad from the perspective of a movie, or just describes some facts without showing any preferences. Four Master's students, which were trained with the background information of the dataset and the basic principles of sentiment analysis, were employed to do the manual checking. Each of them was firstly assigned with a series of MR satisfactions of a subject system for identifying FS. After the first round of checking, the FS results identified by one student were further validated by the other three students. At last, the checking results were inspected by a co-author.

In the evaluation of SA systems, a large number of FS has been discovered. Table 5 reports the FS occurrence rate (namely, R_{FS}) for each MR with respect to the subject systems. As a reminder, a R_{FS} value is calculated as the ratio of the number of FS discovered to the total number of MR satisfactions under investigation, and a higher R_{FS} value indicates that FS occurs more frequently. It can be found that all of the 20 MRs have false satisfactions. Moreover, 43 out of 70 R_{FS} values range from 20% to 50%, while 8 R_{FS} values are higher than 50%.

For the purpose of illustration, we selected some example MGs, on which MT yields varying results for different systems. Table 6 presents MGs of some sentence-level MRs. It can be observed that each of the four subject systems encounters FS for some MRs. Moreover, we have the following findings.

- For some MGs, MT reports satisfactions for all of the four systems, while some of them are FS. As an example for illustration, consider the MG of MR1.1 (on the second row of Table 6). Among the four systems, the Microsoft API produces incorrect source and follow-up outputs that still satisfy MR1.1, while the other three systems all yield correct source and follow-up outputs and accordingly satisfy the MR.
- There are also some MGs for which MT reports satisfactions and violations for different systems, but some of the satisfactions are FS. For example, consider the MG of MR2.2 (on the fourth row of Table 6). MT reports a violation for Microsoft API (since it produces an incorrect follow-up output) and reports satisfactions for the other three systems. However, the Stanford system produces an incorrect follow-up output, and thus this MR satisfaction is a FS.

6.2. *The impact of FS on MT's effectiveness (RQ2)*

It has been shown that applying MT in a user-oriented way can help the users to better understand the capabilities of the target systems so as to be able to make better use of them [13,14]. In the context of SA, different users utilize SA systems for their specific needs, and thus they would concern with the SR values of different MRs. Therefore, the impact of FS on MT's effectiveness is reflected by the differences between information derived from SR_o and SR_g of relevant MRs.

Let us first consider two specific users U_A and U_B who are looking for SA systems with different characteristics: U_A wants to find SA systems that are insensitive to the use of singulars and plurals, while U_B hopes to select SA systems that are insensitive to the order of words connected by 'and'. The SR values of MR1.3 will be of interest to U_A because this MR focuses on investigating how a SA systems handles singulars and plurals. For a similar reason, SR values of MR4.2 will be of interest to U_B. However, SR_o values and SR_g values of these two MRs give totally different recommendations, which are described as below.

(1) *With respect to the* SR_o *values, Heroku has the highest* SR_o *values for both MRs, and thus it's recommended to* U_A *and* U_B. As shown in Table 4, for MR1.3, Heroku has the highest SR_o value (98.99%) among all of the four systems. For MR4.2, the four SR_o values on the four systems are 94.84%, 97.49%, 99.41%, and 90.64%. According to this, both U_A and U_B would decide to use Heroku.
(2) *With respect to* SR_g *values, Amazon is shown to be the best one satisfying both* U_A *and* U_B*'s needs.* As explained in Section 3.2, the SR_g values can be calculated based on the corresponding SR_o values and R_{FS} values. Thus, according to the SR_o values of MR1.3 presented in Table 4 (92.17%, 95.93%, 98.99%, and 86.13%) and the R_{FS} values of MR1.3 reported in Table 5 (11.86%, 31.31%, 33.18% and 42.08%), the four SR_g values of MR1.3 for these four SA systems can be calculated, which are 81.24%, 65.89%, 66.15%, and 49.89%. Note that after considering the FS, all SR

Table 6
Some example MGs that cause false satisfactions: Neg is stand for Negation, and Pos is stand for Positive; □ denotes a true satisfaction, and ◇ represents a true violation.

MR	MG	SA outputs [MT results]							
	(I_s, I_f)	Amazon		Microsoft		Heroku		Stanford	
MR1.1	They're not real, sympathetic, or believable. They are not real, sympathetic, or believable.	Neg (0.96) Neg (0.99)	□	Pos (0.17) Pos (0.17)	[FS]	Neg (0.97) Neg (0.97)	□	Neg Neg	□
MR1.4	It grows tiresome and monotonous. It grows tiresome and monotonous!	Pos (0.59) Pos (0.77)	[FS]	Neg (0.50) Neg (0.62)	□	Neg (1.0) Neg (1.0)	□	Neg Neg	□
MR2.2	Both films are well made with interesting stories set in interesting worlds. Both films are not well made with interesting stories set in interesting worlds.	Pos (0.99) Neg (0.96)	□	Pos (0.50) Pos (0.50)	◇	Pos (1.0) Neg (1.0)	□	Pos Neutral	[FS]
MR3.1	The dvd is a massive achievement. THE DVD IS A MASSIVE ACHIEVEMENT.	Pos (0.98) Pos (0.96)	□	Neutral (0.00) Neutral (0.00)	[FS]	Pos (0.95) Pos (0.95)	□	Pos Neutral	◇
MR3.2	Well, its main problem is that it's simply too jumbled. Well, its MAIN PROBLEM is that it's simply too JUMBLED.	Neg (0.99) Neg (0.97)	◇	Pos (0.08) Pos (0.08)	◇	Neg (1.0) Neg (1.0)	[FS]	Pos Neutral	◇
MR4.3	There's probably a reason for all of this, but we'll never know it. Although there's probably a reason for all of this, we'll never know it.	Neg (0.50) Neutral (0.61)	◇	Neutral (0.00) Neutral (0.00)	□	Neg (0.97) Neg (0.97)	[FS]	Neg Neutral	◇
MR4.4	Making a fictional movie is easier than making one about real life. Making one about real life is harder than making a fictional movie.	Pos (0.69) Neg (0.72)	◇	Pos (0.10) Pos (0.03)	□	Pos (0.96) Pos (1.0)	□	Neutral Neutral	[FS]

values of MR1.3 decrease a lot. More importantly, the preference order among the four systems changes significantly. According to the SR_g values, Amazon (which has the highest SR_g value) rather than Heroku (which has the highest SR_o value) will be recommended to U_A. For MR4.2, the SR_g values are 83.23%, 71.25%, 57.84%, 82.92%, which are again very different from the corresponding SR_o values. Obviously, in this situation, a different system, Amazon, will be selected for U_B (while Huroku was selected based on SR_o values of MR4.2).

A deeper investigation shows that for 16 out of 20 MRs, their SR_o and SR_g suggest different preference orders among the subject systems. Moreover, there are 10 out of 16 MRs that not only suggest different preference orders but also recommend different best performers. This means that FS highly affects MT's effectiveness in testing SA systems.

To summarize, in the evaluation of SA systems, MT results involving no FS (for example, simply using SR_o or VR_o) delivered a quite different conclusion from that taking the FS into consideration (that is, using SR_g or VR_g). However, it is the latter rather than the former that can more accurately reflect the systems' capabilities and performances. Therefore, the existence of FS greatly threatens the effectiveness of MT in testing SA systems, and in turn affects the trustfulness of the evaluation results.

6.3. Factors affecting the occurrence rate of FS (RQ3)

Given the observations of false satisfactions and their impacts on MT, it is natural to consider why there are so many false satisfactions. To this end, we investigated the characteristics of both the SA systems and the MRs, and summarized the following three major factors.

- The root cause for incurring FS is that at least one test input of a non-violating MG is failed. That is, the SA system incorrectly predicts the sentiment information. Hence, the first factor refers to the fact that SA systems frequently produce wrong outputs.
- A notable characteristic of SA's output is that there are only three different sentiment labels. This limited range of output labels affects the possible output relationships of MRs, and also restricts the scope of wrong outputs (for example, if positive is the correct label, then the wrong label can only be negative or neutral). Therefore, the limited range of output labels is another factor.
- The last factor relates to MRs. Although our MRs involve five types of output relationships (as demonstrated in 4), they generally require that the follow-up output has the same (or a different) sentiment label as compared to the source output.

It should be noted that the above three factors collectively magnify the contribution of a large number of FS.

6.4. Threats to validity

With respect to internal validity, when checking for false satisfactions, we have selected only 1% of MR satisfactions. Due to the huge tremendous number of MR satisfactions, only 1% of them still had a lot of MR satisfactions (namely, 7355), even though they might not be representative of all MR satisfactions. However, in order to be safe, we randomly sampled MR satisfactions in terms of individual MRs and subject SA systems, so that we could avoid any possible threats. Therefore, even with this threat, our evaluation results were still very effective in demonstrating high occurrences of FS in the evaluation of SA systems. Another threat to the internal validity is related with the automation of MT. Since the programs required for automation were relatively small-scale and all of them had been carefully reviewed several times, we believe that this threat is low.

Concerning the construct validity, the identification of FS relied on manual checking and thus the corresponding results might not be accurate. To mitigate this threat, we have trained the participants with the background information of the dataset and principles of sentiment analysis. Moreover, three rounds of inspections were conducted in order to avoid personal biases.

The main threat to the external validity is related with the identified MRs. Although 20 MRs were identified for testing SA systems, they might not cover all essential properties of SA systems. However, our MRs considered two typical types of SA systems, namely, sentence-level and document-level SA, and also covered various input operations and output relationships with respect to the application contexts and usages of SA systems.

7. Related work

7.1. Applications of metamorphic testing

In the past decades, MT has been applied to a broad range of domains, including embedded systems [25], web services [45], search engines [12], machine learning [13,27,46], autonomous driving systems [47,48], machine translator [29,30], datalog engines [49], etc. The application of MT has also revealed a series of defects in some typical real-life systems, such as citation systems [6], driverless car system [6], android graphic drivers [50], simulations systems [10,26], and so on.

More recently, MT has been proposed to evaluate SA systems. Ma et al. [19] utilized an MR that inserts or changes sensitive attributes of a sentence without affecting the polarity of the sentence, based on which a testing framework was proposed for detecting fairness violations of SA. Furthermore, Asyrofi et al. [20] proposed BiasFinder, a framework that automatically generates test cases for uncovering SA' bias. BiasFinder leverages an MR to characterize the way of constructing test cases sharing the same prediction results. Another framework, CheckList, used MRs for evaluating SA' robustness as well as its capability of handling some name entities [23]. This study also applies MT for evaluating SA systems. However, we conducted MT in a user-oriented way, and our evaluation aims at revealing to what extent the SA system satisfies the users' specific needs.

7.2. Theories and methods for supporting metamorphic testing

One fundamental problem of MT is the *automatic identification of MRs*. Regarding this issue, various approaches have been proposed, such as the machine learning based MR detection and prediction [51, 52], search based MR inference [53,54], data mutation directed MR acquisition [55], and patterns driven MR identification [5,6]. Furthermore, some tools are developed for MR identification, such as METRIC+ [56], which leverages the category-choice framework for MR identification, and also adopts an output domain-guided mechanism for increasing the efficacy. On the other hand, some factors affecting the effectiveness of MRs have also been studied. For example, Liu et al. [24] empirically investigated the effectiveness of MT, and revealed that a small number of *diverse* MRs are sufficient to achieve satisfactory fault detection effectiveness, while Qiu et al. [57] focused on the *composition* of MRs, and studied the way of constructing effective composite MRs.

Some researchers also studied the way of improving the procedure of MT. Spieker and Gotlieb [58] proposed adaptive MT, which was able to select MRs that are of high fault detection probabilities for the target system by leveraging reinforcement learning technique. Tolksdorf et al. [59] introduced interactive MT to debuggers, where multiple follow-up inputs can be iteratively obtained from an original input.

In contrast to these studies that either focus on the automation of MT or the effectiveness of MT, this study concentrates on the trustfulness of MT results. We identify the impact of false satisfactions on the MT's trustfulness, and also reveal the causes behind the high occurrence rates of FS.

8. Conclusion

Metamorphic testing (MT) has been widely applied for evaluating various applications. In this paper, we have focused on MT's effectiveness with the consideration of false satisfaction for metamorphic relations. A false satisfaction refers to the situation where a metamorphic relation is still satisfied even if some of the relevant outputs are incorrectly computed. With an analysis of false satisfactions, we revealed that without considering false satisfactions, both the violation rate and the satisfaction rate overestimate the system's conformance to the relevant metamorphic relation, and thus it is unable to deliver reliable conclusions about the system capabilities.

In order to empirically investigate the occurrence rate of false satisfactions, we have conducted a study on sentiment analysis. We have also launched experiments to identify false satisfactions in the evaluation results, based on which the impacts of false satisfactions are revealed and the causes behind the high occurrence rates of false satisfactions for the sentiment analysis systems are discussed.

In future research, we plan to launch a larger scale of experiments by analyzing systems from different domains for a more comprehensive picture about the characteristics of false satisfactions for metamorphic relations. On the other hand, similar to the automatic identification of metamorphic relations, the automatic detection of false satisfactions is also important for the overall procedure of MT. Further study shall be conducted on this topic.

CRediT authorship contribution statement

Mingyue Jiang: Methodology, Software, Validation, Visualization, Writing – original draft. **Tsong Yueh Chen:** Conceptualization, Methodology, Writing – review & editing. **Shuai Wang:** Conceptualization, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (Grant No. 61802349 and No. 62132014), and the Zhejiang Provincial Natural Science Foundation of China (Grant No. LY20F020021).

References

[1] T.Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T.H. Tse, Z.Q. Zhou, Metamorphic testing: A review of challenges and opportunities, ACM Comput. Surv. 51 (1) (2018) 4:1–4:27.

[2] S. Segura, G. Fraser, A.B. Sanchez, A. Ruiz-Cortés, A survey on metamorphic testing, IEEE Trans. Softw. Eng. 42 (9) (2016) 805–824.

[3] T.Y. Chen, T.H. Tse, New visions on metamorphic testing after a quarter of a century of inception, in: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2021, 2021, pp. 1487–1490.

[4] X. Xie, J.W.K. Ho, C. Murphy, G. Kaiser, B.W. Xu, T.Y. Chen, Testing and validating machine learning classifiers by metamorphic testing, J. Syst. Softw. 84 (4) (2011) 544–558.

[5] S. Segura, J.A. Parejo, J. Troya, A. Ruiz-Cortés, Metamorphic testing of restful web apis, IEEE Trans. Softw. Eng. 44 (11) (2018) 1083–1099.

[6] Z.Q. Zhou, L. Sun, Metamorphic testing of driverless cars, Commun. ACM 62 (3) (2019) 61–67.

[7] Z.Q. Zhou, T.H. Tse, M. Witheridge, Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors, IEEE Trans. Softw. Eng. 47 (6) (2019) 1164–1183.

[8] A.F. Donaldson, Metamorphic testing of android graphics drivers, in: Proceedings of the 4th International Workshop on Metamorphic Testing, MET '19, 2019, pp. 1–1.

[9] J. Hughes, How to specify it!, in: Proceedings of the International Symposium on Trends in Functional Programming, 2020, pp. 58–83.

[10] X. Lin, M. Simon, N. Niu, Scientific software testing goes serverless: Creating and invoking metamorphic functions, IEEE Softw. 38 (1) (2021) 61–67.

[11] J. Ahlgren, M.E. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, E. Meijer, S. Sapora, J. Spahr-Summers, Testing web enabled simulation at scale using metamorphic testing, in: Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP '21, 2021, pp. 140–149.

[12] Z.Q. Zhou, S. Xiang, T.Y. Chen, Metamorphic testing for software quality assessment: A study of search engines, IEEE Trans. Softw. Eng. 42 (3) (2016) 264–284.

[13] X. Xie, Z. Zhang, T.Y. Chen, Y. Liu, P.-L. Poon, B. Xu, METTLE: a metamorphic testing approach to assessing and validating unsupervised machine learning systems, IEEE Trans. Reliab. 69 (4) (2020) 1293–1322.

[14] Z.Q. Zhou, L. Sun, T.Y. Chen, D. Towey, Metamorphic relations for enhancing system understanding and use, IEEE Trans. Softw. Eng. 46 (10) (2020) 1120–1154.

[15] T.Y. Chen, T.H. Tse, Z.Q. Zhou, Semi-proving: An integrated method for program proving, testing and debugging, IEEE Trans. Softw. Eng. 37 (1) (2011) 109–125.

[16] M. Jiang, T.Y. Chen, F.-C. Kuo, D. Towey, Z. Ding, A metamorphic testing approach for supporting program repair without the need for a test oracle, J. Syst. Softw. 126 (2017) 127–140.

[17] M. Jiang, T.Y. Chen, Z.Q. Zhou, Z. Ding, Input test suites for program repair: A novel construction method based on metamorphic relations, IEEE Trans. Reliab. 70 (1) (2021) 285–303.

[18] Y. Yuan, S. Wang, M. Jiang, T.Y. Chen, Perception matters: Detecting perception failures of vqa models using metamorphic testing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 16908–16917.

[19] P. Ma, S. Wang, J. Liu, Metamorphic testing and certified mitigation of fairness violations in NLP models, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020, pp. 458–465.
[20] M.H. Asyrofi, I.N.B. Yusuf, H.J. Kang, F. Thung, Z. Yang, D. Lo, Biasfinder: Metamorphic test generation to uncover bias for sentiment analysis systems, 2021, arXiv:2102.01859.
[21] J. Ding, T. Wu, J.Q. Lu, X.-H. Hu, Self-checked metamorphic testing of an image processing program, in: 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement, 2010, pp. 190–197.
[22] B. Liu, L. Zhang, A Survey of Opinion Mining and Sentiment Analysis, Springer US, Boston, MA, 2012, pp. 415–463.
[23] M. Ribeiro, T. Wu, C. Guestrin, S. Singh, Beyond accuracy: Behavioral testing of NLP models with CheckList, in: Association for Computational Linguistics (ACL), 2020, pp. 4902–4912.
[24] H. Liu, F.-C. Kuo, D. Towey, T.Y. Chen, How effectively does metamorphic testing alleviate the oracle problem? IEEE Trans. Softw. Eng. 40 (1) (2014) 4–22.
[25] F.-C. Kuo, T.Y. Chen, W.K. Tam, Testing embedded software by metamorphic testing: A wireless metering system case study, in: 2011 IEEE 36th Conference on Local Computer Networks, 2011, pp. 291–294.
[26] M. Olsen, M. Raunak, Increasing validity of simulation models through metamorphic testing, IEEE Trans. Reliab. 68 (1) (2019) 91–108.
[27] J. Ding, X. Kang, X. Hu, Validating a deep learning framework by metamorphic testing, in: 2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET), 2017, pp. 28–34.
[28] S. Wang, Z. Su, Metamorphic object insertion for testing object detection systems, in: 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020, pp. 1053–1065.
[29] P. He, C. Meister, Z. Su, Structure-invariant testing for machine translation, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, ICSE ’20, 2020, pp. 961–973.
[30] Z. Sun, J.M. Zhang, M. Harman, M. Papadakis, L. Zhang, Automatic testing and improvement of machine translation, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, ICSE ’20, 2020, pp. 974–985.
[31] L. Yue, W. Chen, X. Li, W. Zuo, M. Yin, A survey of sentiment analysis in social media, Knowl. Inf. Syst. 60 (2) (2019) 617–663.
[32] M. Hu, B. Liu, Mining and summarizing customer reviews, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 168–177.
[33] G. Mishne, N.S. Glance, Predicting movie sales from blogger sentiment, in: AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs, 2006, pp. 155–158.
[34] X. Li, H. Xie, L. Chen, J. Wang, X. Deng, News impact on stock price return via sentiment analysis, Knowl.-Based Syst. 69 (2014) 14–23.
[35] Y.-M. Li, Y.-L. Shiu, A diffusion mechanism for social advertising over microblogs, Decis. Support Syst. 54 (1) (2012) 9–22.
[36] A. Alamoodi, B. Zaidan, A. Zaidan, O. Albahri, K. Mohammed, R. Malik, E. Almahdi, M. Chyad, Z. Tareq, A. Albahri, H. Hameed, M. Alaa, Sentiment analysis and its applications in fighting covid-19 and infectious diseases: A systematic review, Expert Syst. Appl. 167 (2021) 114155.
[37] A. Hernández, V. Sanchez, G. Sánchez, H. Pérez, J. Olivares, K. Toscano, M. Nakano, V. Martinez, Security attack prediction based on user sentiment analysis of twitter data, in: 2016 IEEE International Conference on Industrial Technology (ICIT), 2016, pp. 610–617.
[38] A. Yousif, Z. Niu, J. Tarus, A. Ahmad, A survey on sentiment analysis of scientific citations, Artif. Intell. Rev. 52 (3) (2019) 1805–1838.
[39] S. Kiritchenko, S. Mohammad, Examining gender and race bias in two hundred sentiment analysis systems, in: Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics, 2018, pp. 43–53.
[40] A. Alhazmi, W.E. Zhang, Q.Z. Sheng, A. Aljubairy, Analyzing the sensitivity of deep neural networks for sentiment analysis: A scoring approach, in: 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–7.
[41] G.A. de Oliveira, R.T. de Sousa, R. de Oliveira Albuquerque, L.J. Garcá Villalba, Adversarial attacks on a lexical sentiment analysis classifier, Comput. Commun. 174 (2021) 154–171.
[42] E.T. Barr, M. Harman, P. McMinn, M. Shahbaz, S. Yoo, The oracle problem in software testing: A survey, IEEE Trans. Softw. Eng. 41 (5) (2015) 507–525.
[43] W. Medhat, A. Hassan, H. Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Eng. J. 5 (4) (2014) 1093–1113.
[44] B. Pang, L. Lee, A sentimental education: Sentiment analysis using subjectivity, in: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL), 2004, pp. 271–278.
[45] P.X. Mai, F. Pastore, A. Goknil, L. Briand, Metamorphic security testing for web systems, in: 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), 2020, pp. 186–197.
[46] A. Chan, L. Ma, F. Juefei-Xu, Y.-S. Ong, X. Xie, M. Xue, Y. Liu, Breaking neural reasoning architectures with metamorphic relation-based adversarial examples, IEEE Trans. Neural Netw. Learn. Syst. (2021) 1–7.
[47] M. Zhang, Y. Zhang, L. Zhang, C. Liu, S. Khurshid, Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, 2018, pp. 132–142.
[48] Y. Tian, K. Pei, S. Jana, B. Ray, Deeptest: Automated testing of deep-neural-network-driven autonomous cars, in: Proceedings of the 40th International Conference on Software Engineering, ICSE ’18, 2018, pp. 303–314.
[49] M.N. Mansur, M. Christakis, V. Wustholz, Metamorphic testing of datalog engines, in: Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2021, pp. 639–650.
[50] A.F. Donaldson, H. Evrard, A. Lascu, P. Thomson, Automated testing of graphics shader compilers, in: Proceedings of the ACM on Programming Languages, 2017, pp. 1–29.
[51] U. Kanewala, J.M. Bieman, Using machine learning techniques to detect metamorphic relations for programs without test oracles, in: 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), 2013, pp. 1–10.
[52] U. Kanewala, J.M. Bieman, A. Ben-Hur, Predicting metamorphic relations for testing scientific software: A machine learning approach using graph kernels, Softw. Test. Verif. Reliab. 26 (3) (2016) 245–269.
[53] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, H. Mei, Search-based inference of polynomial metamorphic relations, in: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ASE ’14, 2014, pp. 701–712.
[54] J. Ayerdi, V. Terragni, A. Arrieta, G. Sagardui, P. Tonella, M. Arratibel, Generating metamorphic relations for cyber–physical systems with genetic programming: An industrial case study, in: Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), no. 1264–1274, 2021.
[55] C.-A. Sun, Y. Liu, Z. Wang, W.K. Chan, μmt: A data mutation directed metamorphic relation acquisition methodology, in: Proceedings of the 1st International Workshop on Metamorphic Testing, MET ’16, 2016, pp. 12–18.
[56] C. Sun, A. Fu, P. Poon, X. Xie, H. Liu, T.Y. Chen, METRIC+: A metamorphic relation identification technique based on input plus output domains, IEEE Trans. Softw. Eng. 47 (9) (2019) 1764–1785.
[57] K. Qiu, Z. Zheng, T. Chen, P.-L. Poon, Theoretical and empirical analyses of the effectiveness of metamorphic relation composition, IEEE Trans. Softw. Eng. (2020) 1.
[58] H. Spieker, A. Gotlieb, Adaptive metamorphic testing with contextual bandits, J. Syst. Softw. 165 (2020) 110574.
[59] S. Tolksdorf, D. Lehmann, M. Pradel, Interactive metamorphic testing of debuggers, in: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, 2019, pp. 273–283.